LA-UR-21-24318
Approved for public release; distribution is unlimited.

Title:	LUNA Condition-Based Monitoring Update: Mahalanobis, SVD, and Auto-Encoder Comparison
Author(s):	Green, Andre Walter
Intended for:	Progress report to sponsor.
Issued:	2021-05-04

LUNA Condition-Based Monitoring Update:

Mahalanobis, SVD, and Auto-Encoder Comparison

Presented 5/4/2021

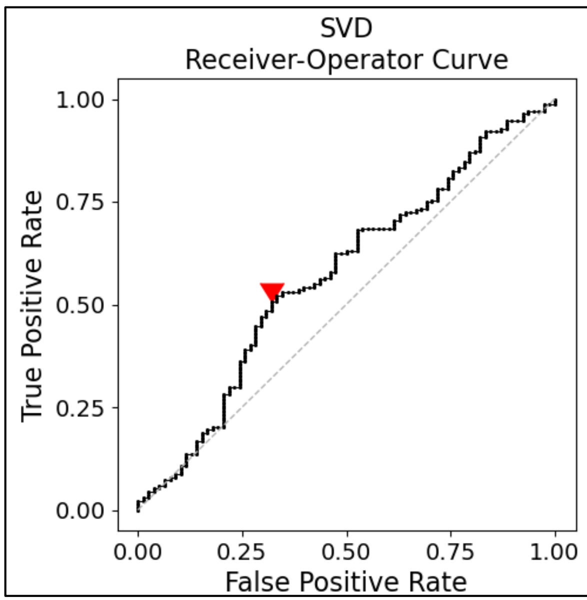
Andre Green

Mahalanobis vs. SVD vs. Auto-Encoder | Data Normalization

The normalization method for the data can have a significant effect on the performance.
5 versions of normalization were tried on this data –

- No normalization
- Bounding each channel to 0 to 1 range
- Standardizing each channel (subtract mean, divide by standard deviation)
- Standardize then bound in 0 to 1 range
- Standardizing then bound in -1 to +1 range
- sklearn's Robust Scaler (medians & quartiles)
- PCA with Whitening

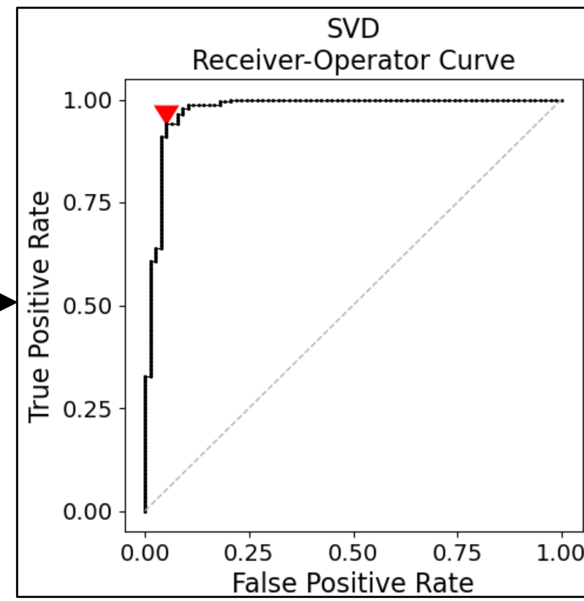
Each statistic (i.e. channel means, channel standard deviations, minima, maxima) were drawn from the undamaged training set used in each fold of the 9-fold cross validation.



Using standardizing normalization on each channel (subtract mean, divide by standard deviation), without bounding channels to a range.

[Comparison_M_S_AE_fold_0_Robust_Scaler.png]

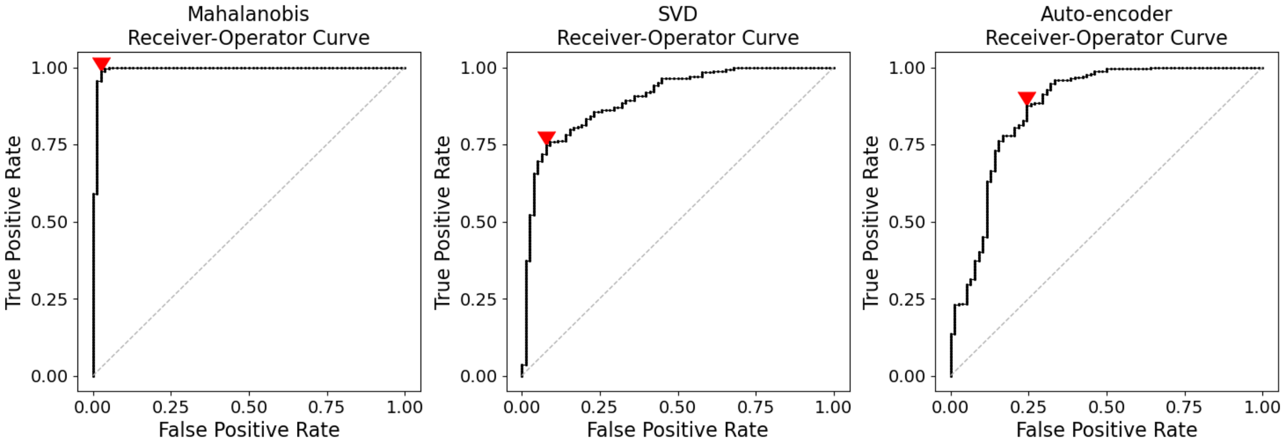
These use the same technique (difference in singular values from SVD), just different normalization.



Using standardizing normalization on each channel, and also bounding to the range -1 to +1.

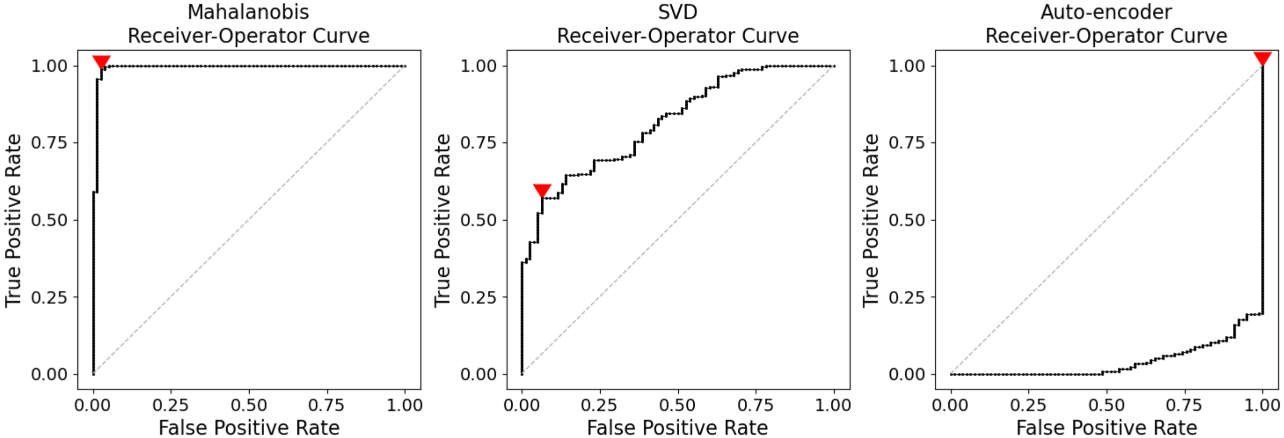
[Comparison_M_S_AE_fold_0_Undamaged_Standardize_and_m1_p1.png]

<u>**Mahalanobis vs. SVD vs. Auto-Encoder**</u> | Data Normalization



Bounding each channel to 0-1 range.

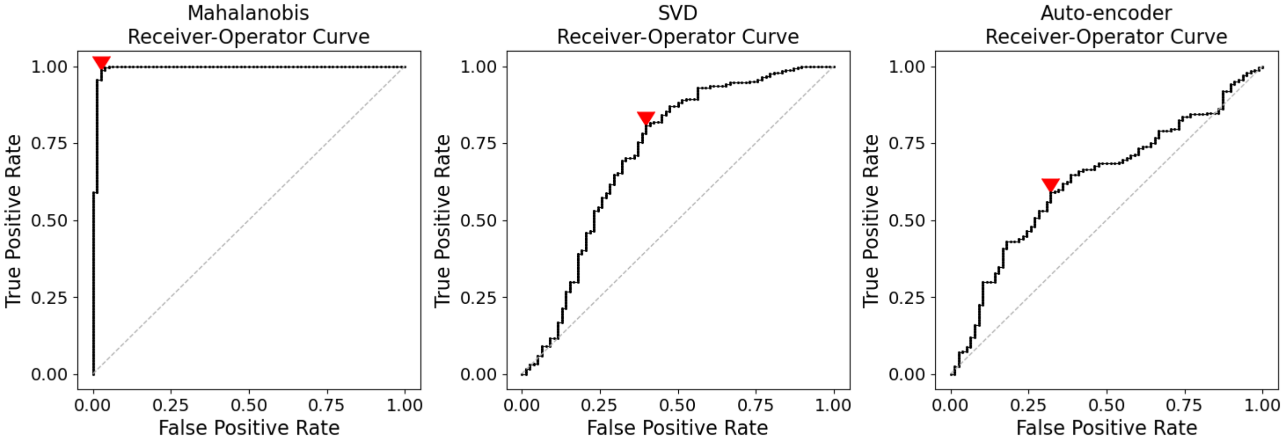
Comparison_M_S_AE_fold_0_Just_01.png



No normalization.

Comparison_M_S_AE_fold_0_No_Normalization.png

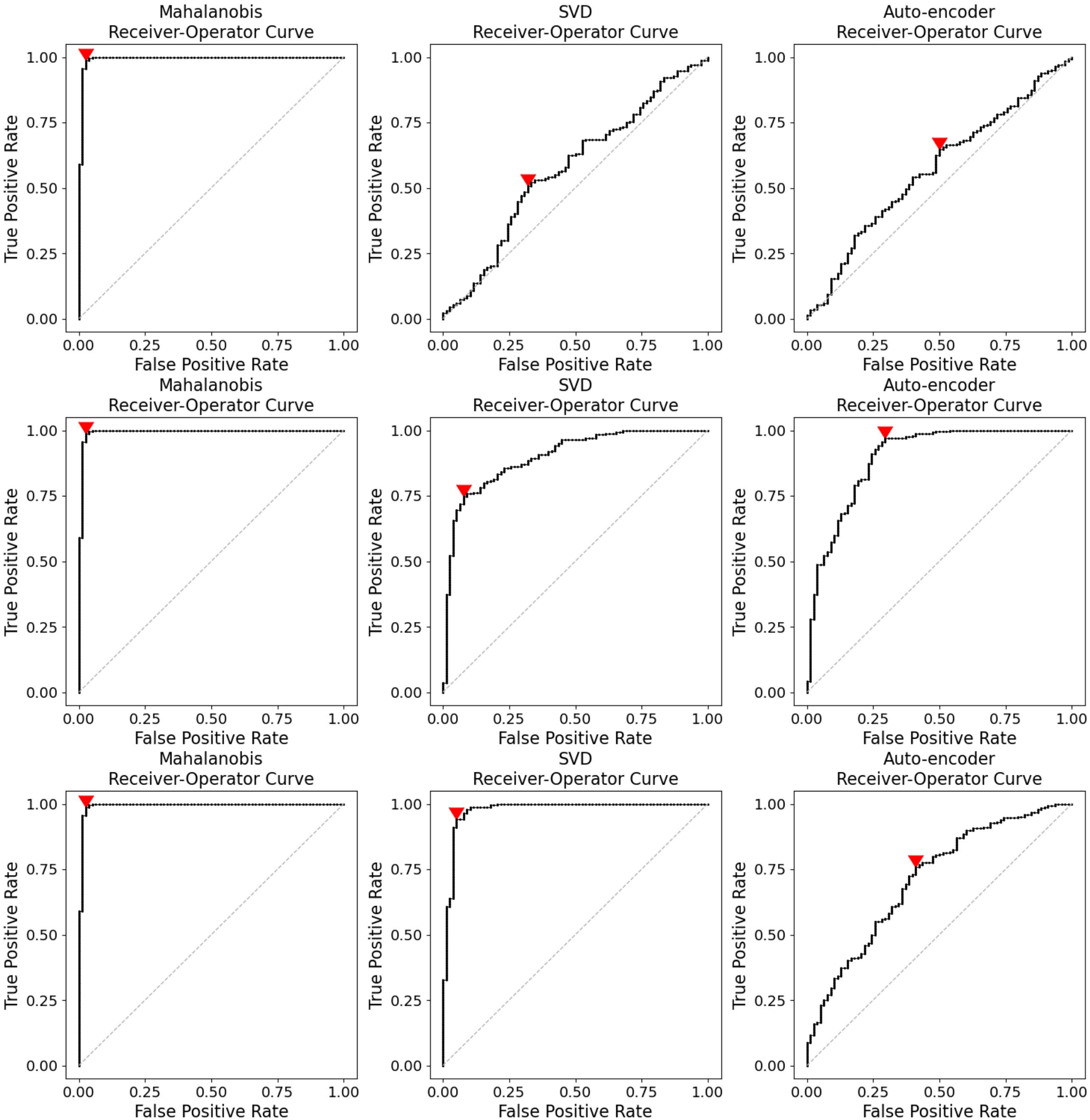
The auto-encoder fails here: this may be because the activation function on the final layer is a sigmoid, which is asymptotically bound to the range [-1, 1].



sklearn Robust Scaler

Comparison_M_S_AE_fold_0_Robust_Scaler.png

Mahalanobis vs. SVD vs. Auto-Encoder | Data Normalization



Standardizing Each Channel

Comparison_M_S_AE_fold_0_Undamaged_Standardize.png

(Subtracting each channel's mean, and then dividing by each channel's standard deviation)

Standardized & Bounded to [0, 1].

Comparison_M_S_AE_fold_0_Undamaged_Standardize_and_01.png

Subtracting each channel's mean, and, dividing by each channel's standard deviation, and then forcing each channel into the range 0, 1. This sets the mean at 0.5 and the training data has no negative values)

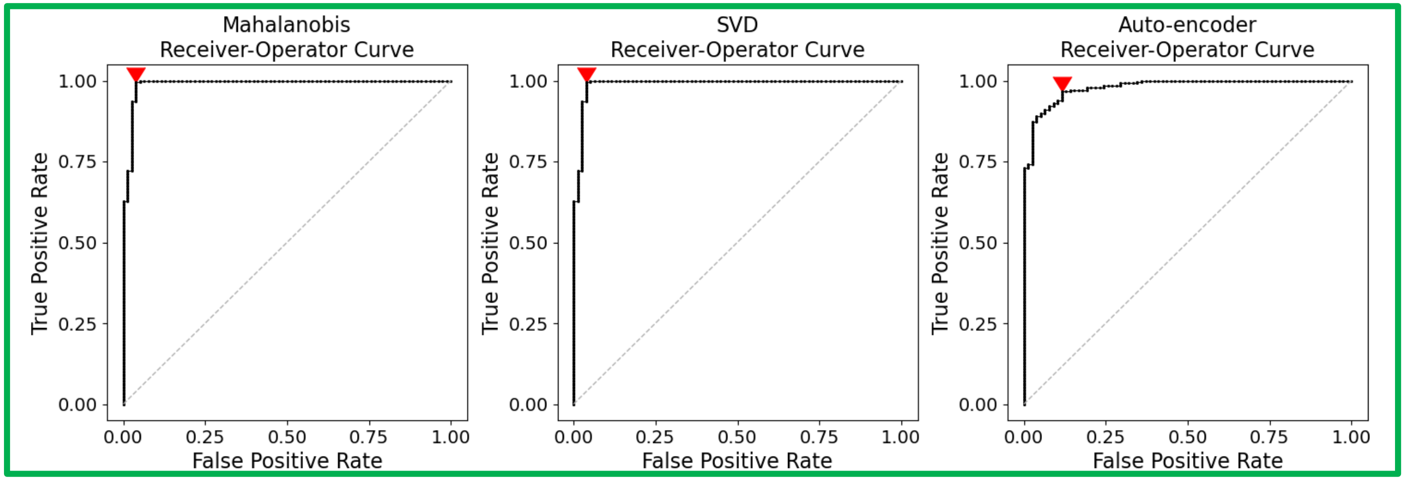
Standardized & Bounded to [-0.5, +0.5].

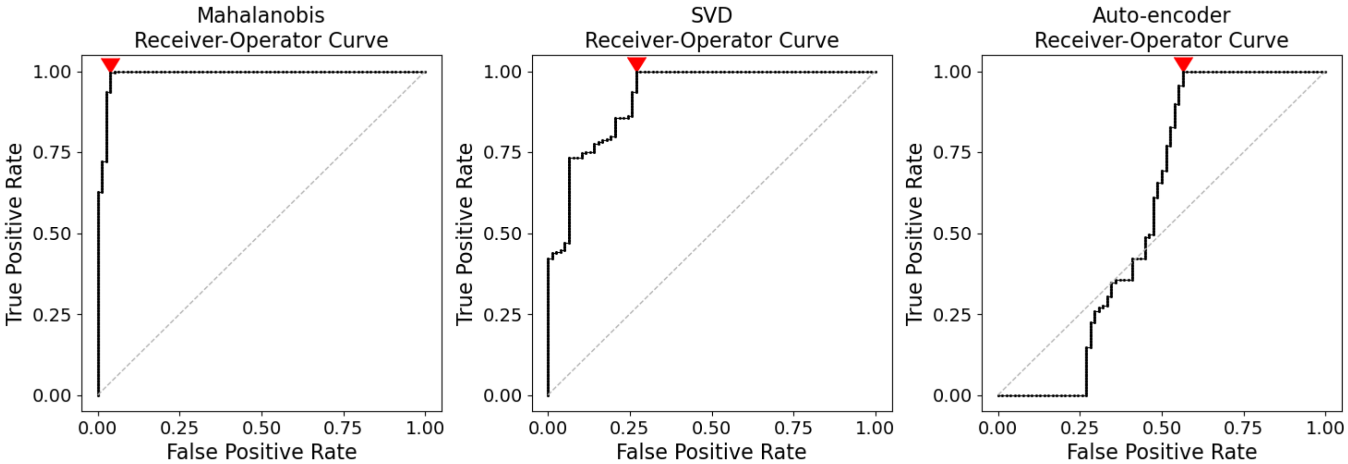
Comparison_M_S_AE_fold_0_Undamaged_Standardize_and_mhalf_phalf.png

Same as above, but forcing each channel into the range 0, 1. This sets the mean at 0, and the training data has some negative values: this appears to work better for the SVD detection method. Same results if [-1, +1] is used.

The auto-encoder uses reLUs internally (whose nonlinearities occur at (0, 0).

Mahalanobis vs. SVD vs. Auto-Encoder | Data Normalization





The Mahalanobis ROC curves are unaffected by the normalization technique used, but for both the SVD-based classifier and the auto-encoder based classifier, Using PCA whitening as a form of normalizing/pre-processing yields the best results (largest difference in true-positive and false-positive rates for a given threshold)

PCA (With Whitening)

The principal components are multiplied by the square root of the number of samples and divided by their corresponding singular value.

In short the data has been projected into an orthogonal basis, and the covariance matrix for the re-projected undamaged training data is the identity.

This doesn't affect Mahalanobis because PCA is ZCA (Mahalanobis), save rotation: https://stats.stackexchange.com/questions/117427/what-is-the-difference-between-zca-whitening-and-pca-whitening

https://stats.stackexchange.com/questions/166525/is-mahalanobis-distance-equivalent-to-the-euclidean-one-on-the-pca-rotated-data

PCA (No Whitening)

The data are projected into an orthogonal basis, but the components are not scaled based on their singular values.

"A model with large weight values is often unstable, meaning that it may suffer from poor performance during learning and sensitivity to input values resulting in higher generalization error."

https://machinelearningmastery.com/how-to-improve-neural-network-stability-and-modeling-performance-with-data-scaling/

Axes for ELoad

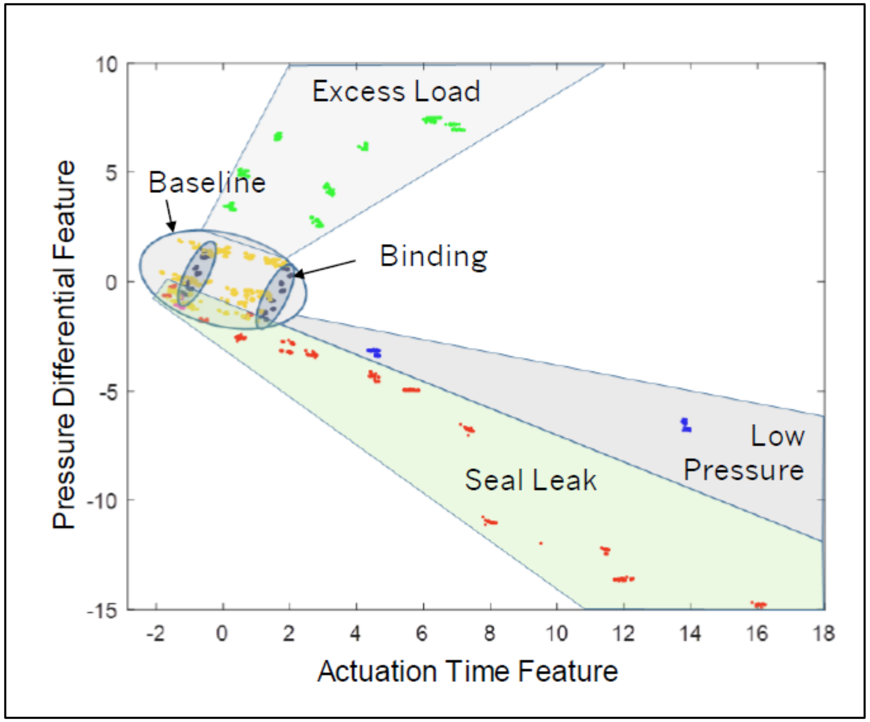
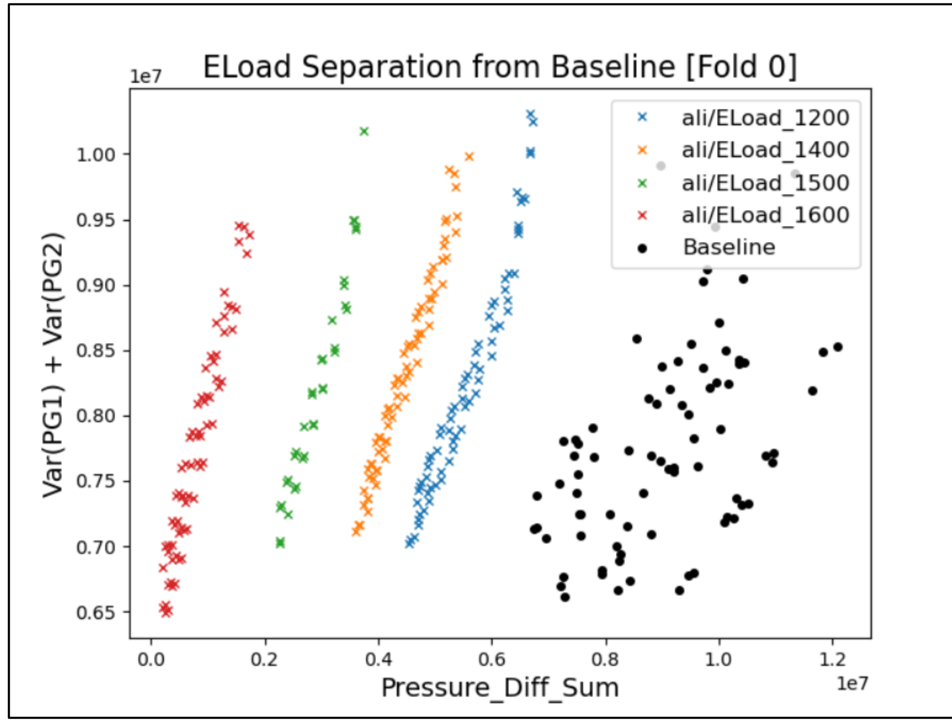


Fig. 9 from 'Embedded Sensing System for Condition Monitoring of Hydraulic Actuators' shows the excess load being separated by the pressure differential.

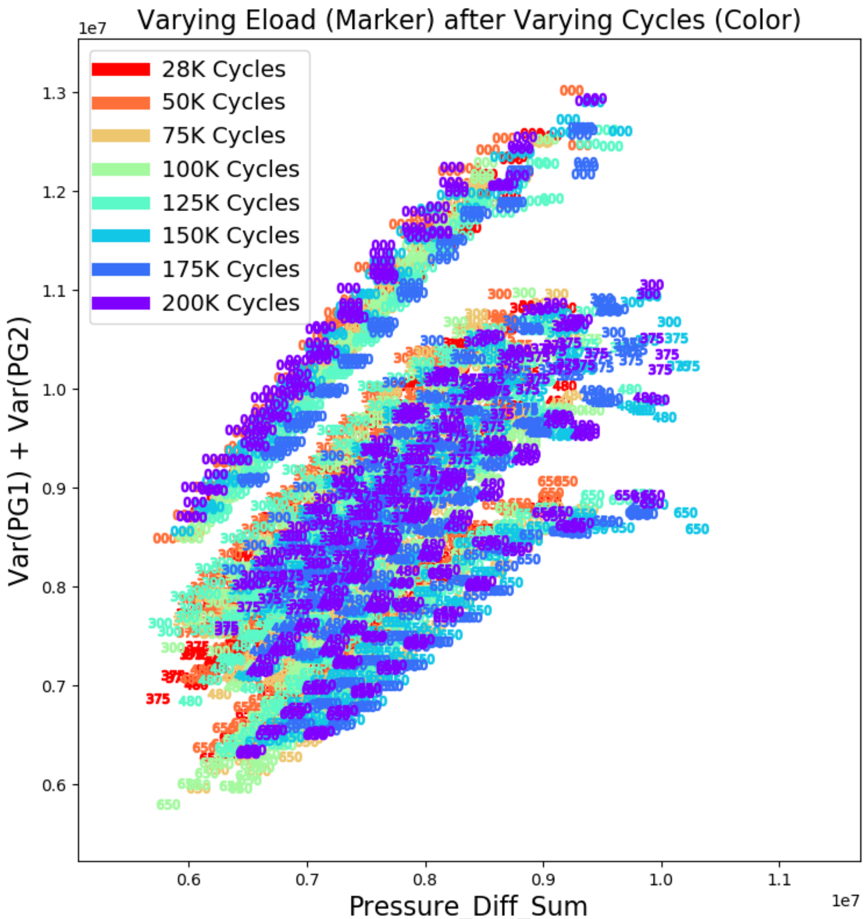


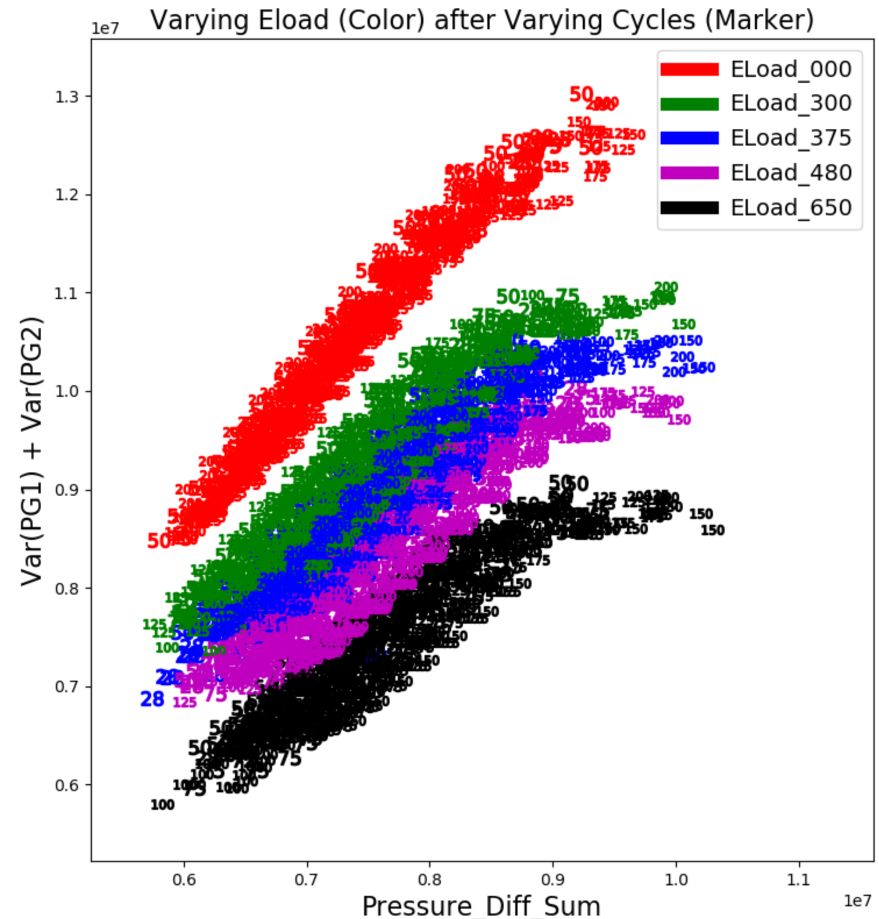
The pressure differential helps separate the ELoad cases for this dataset too. The sum of the variance in the pressures is helpful in separating some Eload1200 cases from the Baseline.

The axes were acquired by examining the top components of PCA for the dataset (just Eload and Baseline) and maximizing the number of zeroed entries.

Higher Cycles Data

The graphs below use the data collected after 28 – 200K cycles.

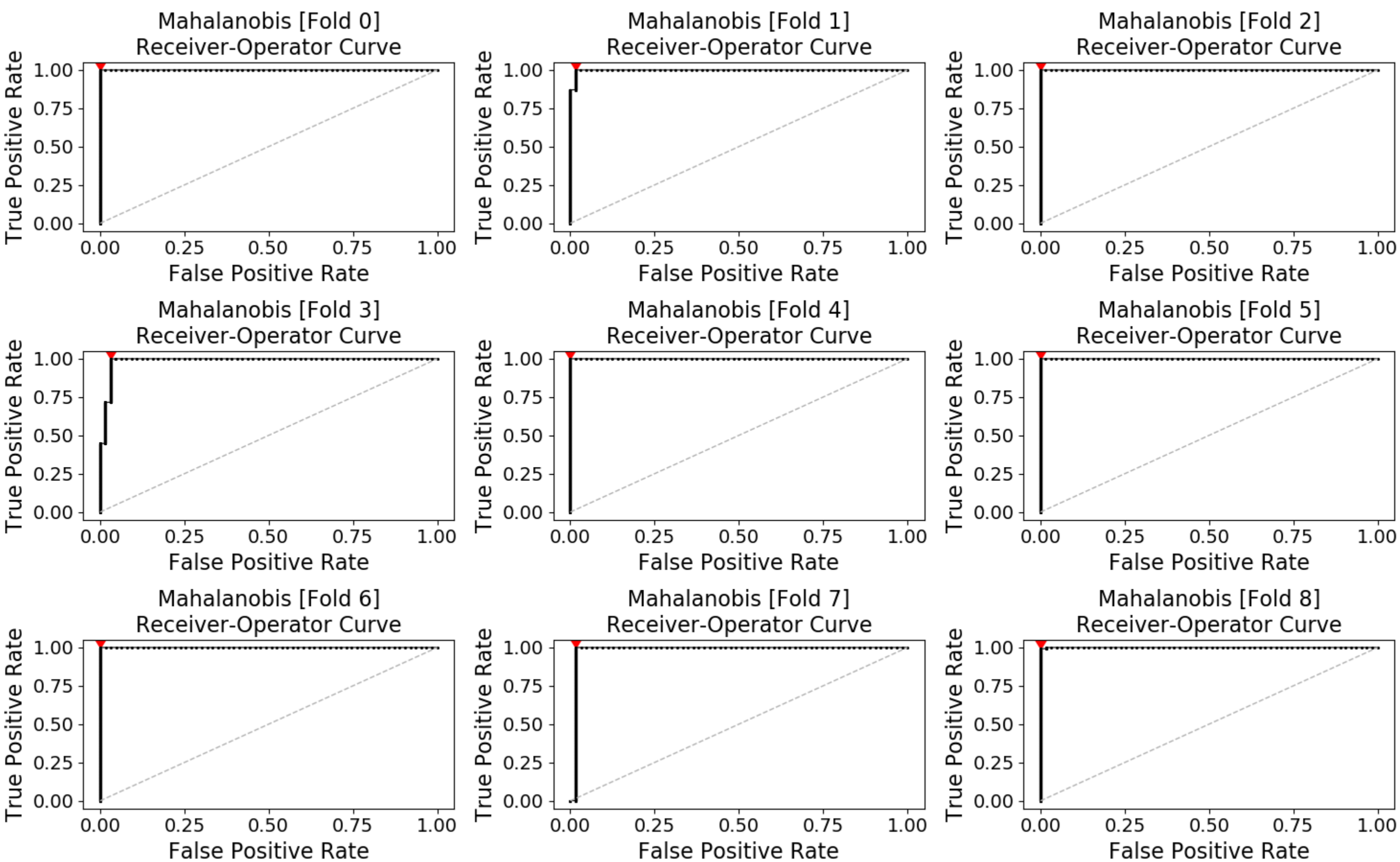


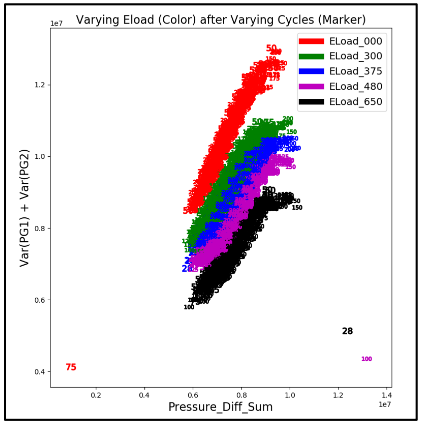


Both graphs show the same data. The graph on the left colors by *the number of cycles*, and the graph on the right colors by *the severity of the damage*.

The left graphs suggests that over many actuation cycles that both the sum variances in pressure and the sum of the difference between the pressures increases.

Mahalanobis Distance on Higher Cycles Data





All 9 folds have very good ROC curves (most have 0% false positives and 100% true positives). Fold 3 has the most trouble. Interestingly, the outlier points (shown on right) are never misclassified with respect to damage.

For Fold 1, the false positive is a fairly normal looking point – no features on it are more than 1 standard deviation, it's presumably just because a large number of its features deviate from the mean a moderate amount.

For Fold 3, the false positives have very unusual 'Diff_Temp_Var' values (>10 STDVs from mean)
For Fold 7, the false positive has a very unusual accelerometer variances (>20 STDVs from mean)

Mahalanobis Distance on Higher Cycles Data

Fold 0:	**Test acc: 100.000%**	|	Train acc: 100.000%
Fold 1:	**Test acc: 100.000%**	|	Train acc: 99.917%
Fold 2:	**Test acc: 99.834%**	|	Train acc: 100.000%
Fold 3:	**Test acc: 99.834%**	|	Train acc: 99.834%
Fold 4:	**Test acc: 100.000%**	|	Train acc: 100.000%
Fold 5:	**Test acc: 99.917%**	|	Train acc: 100.000%
Fold 6:	**Test acc: 100.000%**	|	Train acc: 100.000%
Fold 7:	**Test acc: 99.917%**	|	Train acc: 100.000%
Fold 8:	**Test acc: 99.086%**	|	Train acc: 99.418%

Why not just use the ROC curve?

The additional step of using half the testing data to determine the threshold (from the ROC curve) and the remaining half to test the accuracy is a better representation of how accurate the technique really is.

The ROC curve can be used to select the threshold for a classifier – but re-using that threshold on the data from which it was drawn does not tell you anything about how well it will perform on unseen data.

Performance

At worst, the Mahalanobis distance had 99.83% accuracy in determining whether a given sample was damaged or not. (This is approximately 1-2 misclassifications per thousand)

Training/Testing Procedure

Using 9-fold validation, 8/9ths of the undamaged data is used to construct the inverse covariance matrix and mean vector used for Mahalanobis.

Half of the remaining 1/9th of the undamaged data and the damaged data is used to determine the threshold for the Mahalanobis curve (i.e. the ideal threshold on the ROC curve) to optimize accuracy, and the remaining half is used to test the Mahalanobis damage classifier.